X

Most Useful PowerShell Cmdlets for Managing and Securing Active Directory

Register Now

sponsored by netwrix

About Advertise Forums Log In

Petri
IT Knowledgebase

Microsoft 365

Now is the time to shift Shift to Microsoft 365 >

PETRI NEWSLETTER SIGN-UP

TECH TUESDAY

Subscribe to Tech Tuesday, the latest insights from Petri.com for IT Pros.

Email

Creating Your First PowerShell Class

Posted on July 28, 2017 by **Missy Januszko** in PowerShell with 1 Comment

WE USE COOKIES TO IMPROVE YOUR BROWSING EXPERIENCE. Find out more about our cookie policy *here*.

Share Tweet Share

PowerShell 5.0 introduced the concept of being able to create classes directly from within PowerShell. Prior to version 5.0, you needed to define a class in C# and compile it, or use some pretty complicated PowerShell to create one. For PowerShell scripters without a programming background, you may be wondering what all the fuss around classes is about. You may also be thinking, "so what?" You may feel intimidated on getting started with classes. This article intends to take the fear out of using classes in your PowerShell programming.

[Sponsored]

What Is a Class, Exactly?

Wikipedia states, "In object-oriented programming, a class is an extensible program-code-template for creating objects, providing initial values for state (member variables), and implementation of behavior (member functions of methods)." Huh? Well, wait a second. First, a **class** is a template for creating **objects**. You have been using PowerShell to create and manipulate objects all along. Next, the class template provides initial values for its **members**. PowerShell objects have members too, properties and methods, and you can provide values for the properties. Lastly, the class implements behavior via **methods**. So does PowerShell!

You Have Been Using Classes All Along

Take a moment to reflect on what happens when a PowerShell cmdlet runs. For example, when the get-service command runs, it returns an object. By running get-service and then piping it to get-member, we find out that the output of get-service is an object of type System.ServiceProcess.ServiceController. A quick MSDN lookup reveals that ServiceController is a .Net class. Likewise, for many PowerShell cmdlets, they return an instance of a class.

Functions, Classes, What Is the Difference?

Microsoft invented PowerShell for Windows administrators to simplify and automate administrative tasks. Its intent was to be user-friendly for everyone, regardless of previous programming experience. In addition, it simplifies the transition from using the GUI to using code with its easy-to-read language. However, PowerShell classes appeal to programmers who already have experience using classes in other languages. That is, Microsoft is expanding the intended audience of developers to admins. For the admins already using PowerShell, classes can ease them from “scripter/admin” to “developer” by investigating and understanding the similarities between the two.

How Do I Start Building a Class?

I am going to start building a sample class called “Rock”. Think of an item you find in your yard or on the street. Where to start? Remember writing your first function? You simply used the Function {} keyword and then built the function from there. Building a class just uses a different keyword, the Class {} keyword.

Adding Properties to your Class

I do not know much about rocks, so I first started contemplating the properties of rocks. Rocks can be different colors, shapes, and sizes. Beyond that, I used my google-fu to find out more about the properties of rocks. I included color, luster, shape, texture, and pattern from the linked reference. I also added size and location as more information I would like to have on an instance of a rock. I am going to use strings for the type of each of these properties except for the location property. I am defining the location property as an integer, distance from the person holding the rock, perhaps.

PowerShell
```
class Rock {
    [string]$Color
    [string]$Luster
    [string]$Shape
    [string]$Texture
    [string]$Pattern
    [string]$Size
    [int]$Location
    }
```

Class Properties — Similar to Function Parameters

For a programming comparison, maybe you are creating a SMBShare class. You do not need to Google or Bing to know that a SMBShare has a name, a path, and permissions. If you were creating a function to manipulate an SMBShare object, these would be the parameters you would define. Thus, **properties** of a class loosely translate to **parameters** of a function.

Big Words, Coming Up!

Programming books like to use big words to talk about how to create a class. The first word you will see often is **instantiation** but let’s break this word down. Instantiation is the noun form of the verb “instantiate,” which means “to create an instance of.” You can even see the root word “instance” in both instantiate and instantiation. To instantiate an object of the rock class, I am merely creating an instance of a rock.

Creating the Instance of the Object

It is time to instantiate a Rock object. To do so, I need to introduce the second big word, **constructor**. A constructor is a “special type of subroutine that is called to create an object.” The constructor is named the same as the class. You use the new() operator to create (or instantiate!) a new rock object. Programmatically, it looks like this:

PowerShell
```
$rock = [rock]::New()
```

I am creating a new object of the [rock] class and storing it in the variable $Rock. From there, I can display the contents of $Rock or pipe $Rock to get-member to check out its type, properties, and methods.

Overcoming Your Fear of Using Classes

I have not written any code, yet. So far, I have just designed what my Rock class should look like, what its properties are, and created a new instance of a Rock using the constructor. Now that I have $Rock, I assign values to each of the properties just like I would in any ordinary PowerShell function.

There are not any new, special words to describe this action. I am simply assigning values to the properties as I would in creating an SMBShare or gathering information using WMI and creating a custom object with the properties I want. And although I introduced a couple of new programming concepts, there are not a ton of differences between creating a Rock from a class definition and creating a custom object.

I hope you will challenge yourself and give it a try.

Tagged with **Automation**, **Beginner**, **Classes**, **PowerShell**, and **Scripting**

1 Comments

Sort by Votes | Date

silverback - 4 months ago

Man, thanks a bunch! It is an extremely usefull explanation.

0 |